10 轮 Midori128 的中间相遇攻击 *

刘　亚 [1a, 1b, 2†]，刁倩倩 [1a, 3]，李　玮 [4, 5]，刘志强 [2]

(1. 上海理工大学 a. 光电信息与计算机工程学院, b. 上海市现代化光学重点实验室, 上海 200093; 2. 上海交通大学 计算机科学与工程系, 上海 200240; 3. 上海观源信息科技有限公司, 上海 200240; 4. 东华大学 计算机科学与技术学院, 上海 201620; 5. 上海市信息安全综合管理技术研究重点实验室, 上海 200240)

摘　要：轻量级分组密码由于软硬件实现代价小且功耗低，被广泛地运用资源受限的智能设备中保护数据的安全。Midori 是在 2015 年亚密会议上发布的轻量级分组密码算法，分组长度分为 64 bit 和 128 bit 两种，分别记为 Midori64 和 Midori128，目前仍没有 Midori128 抵抗中间相遇攻击的结果。通过研究 Midori128 算法基本结构和密钥编排计划特点，结合差分枚举和相关密钥筛选技巧构造了一条 7 轮中间相遇区分器。再在此区分器前端增加一轮，后端增加两轮，利用时空折中的方法，提出对 10 轮的 Midori128 算法的第一个中间相遇攻击，整个攻击需要的时间复杂度为 $2^{126.5}$ 次 10 轮 Midori128 加密，数据复杂度为 2^{125} 选择明文，存储复杂度 2^{105} 128-bit 块，这是首次对 Midori128 进行了中间相遇攻击。

关键词：分组密码；中间相遇攻击；Midori128

中图分类号：TP309.2　　doi: 10.3969/j.issn.1001-3695.2017.07.0701

Meet-in-the-middle attacks on 10-round Midori128

Liu Ya[1a, 1b, 2†], Diao Qianqian[1a, 3], Li Wei[4, 5], Liu Zhiqiang[2]

(1. a. *School of Optical-Electrical & Computer Engineering*, *b. Engineering Research Center of Optical Instrument & System*, *Ministry of Education*, *Shanghai Key Lab of Modern Optical System*, *University of Shanghai for Science & Technology*, *Shanghai* 200093, *China*; 2. *Dept. of Computer Science & Engineering*, *Shanghai Jiao Tong University*, *Shanghai* 200240, *China*; 3. *Shanghai Viewsource Information Science & Technology Co, Ltd*, *Shanghai* 200240, *China*; 4. *School of Computer Science & Technology*, *Donghua University*, *Shanghai* 201620, *China*; 5. *Shanghai Key Laboratory of Integrate Administration Security*, *Shanghai* 200240, *China*)

Abstract: The lightweight block ciphers can be widely used in various applications, such as smart cities, internet of things and cloud computation and so on, in order to protect data and information secure. Midori is a lightweight block cipher proposed in ASIACRYPT 2015. Its block size has two scenarios, i. e, 64 bits and 128 bit, denoted by Midori64 and Midori128 respectively. Up to now, there are no results about meet-in-the-middle attacks on Midori128. This paper developed a meet-in-the-middle attack on 10-round Midori128 for the first time. Specifically, studying the basic construction and key schedule of Midori128, this paper constructed a 7-round distinguisher on Midori128 by using the differential enumeration and key-dependent sieve techniques. Through appending one round at its top and two rounds at its bottom, this paper mounted a meet-in-the-middle attack on 10-round Midori128. In the attack, time-memory tradeoff technique and some weak subkeys were considered so as to reduce the time complexity of online phase. Finally, the data, time and memory complexities of our attack are 2^{125} chosen plaintexts, $2^{126.5}$ 10-round encryptions and 2^{105} 128-bit blocks, respectively.

Key Words: block cipher; meet-in-the-middle attack; midori128

0　引言

随着互联网技术的发展和智慧城市的建设，存储和计算资源受限的设备如 RFID、传感器等在物联网、云计算以及智慧城市中运用的越来越广泛，如何保证这些设备中数据的安全是一个重要的研究课题。轻量级分组密码由于其软硬件实现效率快、功耗低等特点被广泛地运用到资源受限的设备中保护数据的安全，但高的实现效率必然会牺牲一部分安全性，因此研究轻量级分组密码算法抵抗各种有效的攻击手段是十分必要的。2015 年，Banik 等人[1]在亚密上发布了一个新的轻量级分组密码算法——Midori。该算法基于 SPN 结构，轮数为 20，密钥长度为 128 bit，分组长度分为 64 bit 和 128 bit 两种。根据分组长度不同，

基金项目：国家自然科学基金资助项目（61402288，61772129，61672347，61472250）；上海市自然科学基金资助项目（15ZR1400300）；上海市教育委员会科研创新重点项目（14ZZ066）；闵行区产学研合作计划项目（2016MH310）；上海市信息安全综合管理技术研究重点实验室开放课题（AGK201703）

作者简介：刘亚（1983-），女（通信作者），安徽当涂人，讲师，工学博士，主要研究方向为信息安全、密码学等（liuya@usst.edu.cn）；刁倩倩（1992-），女，江苏沭阳人，硕士研究生，主要研究方向为分组密码安全性分析；李玮（1980-），女，安徽淮南人，副教授，工学博士，主要研究方向为信息安全、密码学等；刘志强（1976-），男，江西上饶人，工学博士，主要研究方向为信息安全、密码学、区块链等.

将 Midori 算法记为 Midori64 和 Midori128。

Midori 算法由于功耗低、实现效率高，在现实生活中具有广泛的应用前景，而研究它的安全性可以为未来在现实中使用提供理论依据。目前，Midori64 的安全性已经被差分攻击、不可能差分分析、相关密码不可能差分分析、高阶差分分析、中间相遇攻击以及不变子空间攻击评估过[2~6]，Midori128 的安全性仅被差分分析、不可能差分分析和相关密钥差分分析评估过[7~10]。文献[7,8]利用不可能差分分析分别攻击了不考虑白化密钥情况下的 10 轮 Midori128 和 11 轮 Midori128；文献[9]运用了差分分析方法攻击了 13 轮 Midori128；文献[10]给出了 Midori128 全轮的相关密钥差分分析。但目前国内外研究者还没有评估过 Midori128 抵抗中间相遇攻击的能力，而中间相遇攻击是近年来十分有效的攻击手段，被用来分析过许多著名密码算法的安全性并得到该算法最好的分析结果，如 AES 算法[11]等，因此研究 Midori128 抵抗中间相遇攻击也是重要的。

中间相遇攻击最早由 Diffie 等人[12]提出，其主要思想是将分组密码分为两个部分或者三个部分（中间部分为区分器），然后猜测两端的相关密钥来加/解密一系列的明密文。若得到的中间状态值相等或满足区分器的预计算表，则此猜测密钥被保留，否则被排除。通过一系列明密文最后可以筛选出正确的密钥。

在仔细研究了 Midori128 算法结构和密钥编排计划的基础上，首次提出了对 10 轮 Midori128 算法中间相遇攻击。在攻击过程中，利用差分枚举技术和相关密钥筛选技术构造了一个 7 轮的中间相遇区分器，再在区分器前端增加一轮，后端增加二轮，首次实现了对 10 轮 Midori128 算法的中间相遇攻击。

1 预备知识

1.1 符号标记

在介绍算法之前，先列出符号标记，具体如下：

a) *P, C:* 明文，密文；

b) $\square\square_{\square}$: 轮密钥

c) x_i, y_i, z_i, w_i: 分别表示在第 i 轮的单元替换、单元混合、列混淆和密钥加操作之前的中间状态值;

d) $x_i[j]$：表示 i 轮的第 j 个单元；

e) $\square_{\square}^{\square}[\square]$: 表示一系列$\square_{\square}[\square]$的第 k 个取值；

f) $\Delta\square_{\square}^{\square}[\square]$: $\Delta\square_{\square}^{\square}[\square] = \square_{\square}^{\square}[\square]\oplus\square_{\square}^{0}[\square]$;

g) $\square\square_{\square}$: $\square\square_{\square} = \square\square\square\square\square\square\square\square\square\square^{-1}(\square\square_{\square})$;

h) $\bar{\square}_{\square}$: $\bar{\square}_{\square} = \square\square\square\square\square\square\square\square\square\square^{-1}(\square_{\square+1})$。

1.2 Midori 算法描述

Midori 算法是 2015 年亚密会议上由 Banik 等人发布的基于 SPN 结构的轻量级分组密码算法，每一轮的轮函数包括四种运算，分别为单元替换(SubCell)、单元混合(ShuffleCell)、列混淆(MixColumn)和密钥加(KeyAdd)，如图 1 所示。其中在第一轮加密之前对明文有一个轮密钥加操作，最后一轮取消了单元混合和列混淆操作，只包括单元替换和密钥加操作。

将 Midori 算法的状态表示为如下的 4×4 矩阵，每一个单元是 8 bit。

$$S = \begin{pmatrix} s_0 & s_4 & s_8 & s_{12} \\ s_1 & s_5 & s_9 & s_{13} \\ s_2 & s_6 & s_{10} & s_{14} \\ s_3 & s_7 & s_{11} & s_{15} \end{pmatrix}$$

单元替换(SubCell)：每一个字节运用非线性 S 盒作单元替换操作。

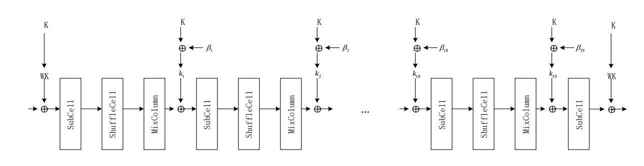


图 1 Miori128 算法构造

单元混合(ShuffleCell)：按字节做移位操作。

$(\square_0,\square_1,\square_2,\square_3,\square_4,\square_5,\square_6,\square_7,\square_8,\square_9,\square_{10},\square_{11},\square_{12},\square_{13},\square_{14},\square_{15})$

$\leftarrow (\square_0,\square_{10},\square_5,\square_{15},\square_{14},\square_4,\square_{11},\square_1,\square_9,\square_3,\square_{12},\square_6,\square_7,\square_{13},\square_2,\square_8)$

列混淆(MixColumn)：每一列乘以4×4的 M 矩阵运算。

$$M = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

密钥加(KeyAdd)：将轮密钥和 S 作异或操作。

密钥编排算法： Midori128 算法第一轮的白化密钥和最后一轮的子密钥是 128bit 主密钥。中间轮密钥为主密钥与一个常数异或，即$rk_i = K_i \oplus \beta_i (1 \le i \le 19)$，其中$\beta_i$为常数。

定义 1(2-δ-set)[13] 2-δ-set 将一个状态的 2 个字节（活动单元）取任意值，其他 14 个字节（固定单元）取固定值所得到的一个集合，此集合共有$2^{2\times8}$个元素。

命题 1(S 盒的差分特性) 对于一个给定的 S 盒，若输入差分为Δ_i，输出差分为Δ_o，则平均有一个输入值x满足$S(x)\oplus S(x\oplus\Delta_i) = \Delta_o$。

2 中间相遇攻击简介

中间相遇攻击最早是由 Diffie 和 Hellman 提出的，被广泛应用于 Hash 函数和分组密码尤其是 AES 算法的安全性分析中[14]。它的基本原理有两种：第一种将加密算法E分解成$E = E_2 \circ E_1$，其中E_1和E_2部分的密钥分别为k_1和k_2，猜测k_1和k_2的值，若对某个明密文对(P, C)，使得$E_1(P, k_1) = E_2(C, k_2)$，则$k_1$和$k_2$的猜测值是正确的，否则即为错误的密钥；第二种将一个加密算法E分解成三个部分$E = E_2 \circ E_{min} \circ E_1$ ，其中E_{min}中存在一条多轮的区分器，E_1和E_2部分的相关密钥分别为(k_1, k_2) ，首先在E_{min}部分预计算一个有序序列，然后对某个明密文对(P,C)，猜测k_1和k_2的值，若$E_1(P, k_1)$和$E_2(C, k_2)$满足预计算的有序序列，则此猜测密钥是正确的，否则被淘汰。

2000 年，Gilbert 等人[15]利用 4 轮的 AES 区分器进行 7 轮碰撞攻击。2008 年，Demirci 等人[16]将 AES 中间相遇区分器扩

展为 5 轮，并改进 8 轮的 AES-256 的分析结果。随后，在 2010 年，Dunkelman 等人[17]提出了差分枚举技巧等几个降低预计算参数的方法，改进 7/8 轮 AES-192/256 的中间相遇分析结果。2014 年，Li 等人[18]利用密钥编排规律，改进了 5 轮 AES 的中间相遇区分器，并给出了 9 轮的 AES-192 的中间相遇攻击。2016 年，文献[11]利用了差分枚举技巧和依赖密钥筛选技巧来构造 6 轮中间相遇区分器，进一步降低预计算参数，并改进 10 轮 AES-256 的分析结果。此外，文献[19~22]也给出了 LED 、CLEFIA、 Camellia、TWINE、LBlock 算法的中间相遇分析。

3　10 轮 Midori128 算法的中间相遇攻击问题

首先运用差分枚举和密钥筛选策略构造一个 7 轮中间相遇区分器，如图 2 所示，然后在此区分器的前端增加 1 轮，后端增加 2 轮实施了对 10 轮 Midori128 算法的中间相遇攻击。

3.1　10 轮 Midori128 中间相遇区分器

在图 2 中，因为$\square_8[5]=\square_8[4]\oplus\square_8[6]\oplus\square_8[7]$，$\square_8[6]=\square_8[4]\oplus\square_8[5]\oplus\square_8[7]$，所以$\square_8[5]\oplus\square_8[6]=\square_8[6]\oplus\square_8[5]$。令$\square_{\square\square}=\square_8[6]\oplus\square_8[5]$，$\square_{\square\square\square}=\square_9[5]\oplus\square_9[6]$，则$\square_{\square\square\square}=\square_{\square\square}\oplus\square\square_8[5]\oplus\square\square_8[6]$。

命题 2　设$\square_2[3,12]$为活动字节，则$\{\square_2^0,\square_2^1,\cdots,\square_2^{2^{16}-1}\}$构成了一个 2-δ-set。取$\square_2^0$，然后再依次选取 18 个$\square_2^{\square}(\square=1,\cdots,18)$使得其经过 S 盒运算后满足$\Delta\square_2^{\square}[3]=\Delta\square_2^{\square}[12]$，将这 19 个值经过 7 轮加密，若此集合中一个元素满足图 2 中的截断差分，则与 19 个值相关的 144 bit 的有序序列$(\square^1_{\square\square\square}\oplus\square^0_{\square\square\square},\square^2_{\square\square\square}\oplus\square^0_{\square\square\square},\cdots,\square^{18}_{\square\square\square}\oplus\square^0_{\square\square\square})$，共$2^{136}$个值。

证明　如图 2 所示，当一个消息对$(\square_2^{\square},\square_2^{\square})$满足截断差分链时，那么经过 7 轮加密后，输出差分序列$(\square^1_{\square\square\square}\oplus\square^0_{\square\square\square},\square^2_{\square\square\square}\oplus\square^0_{\square\square\square},\cdots,\square^{18}_{\square\square\square}\oplus\square^0_{\square\square\square})$的值是由如下 46 个字节决定的：

$$\square_2^{\square 0}[3,12]\parallel\square_3^{\square 0}[9,10]\parallel\square_4^{\square 0}[0,2,3,9,10,11]$$
$$\parallel\square_5^{\square 0}[0,1,2,3,4,5,7,8,9,10,11,12,13,15]$$
$$\parallel\square_6^{\square 0}[0,1,2,3,4,5,6,8,9,10,11,12,13,15]$$
$$\parallel\square\square_6[1,3,4,9,11,12]\parallel\square\square_7[4,11]$$

可以进一步将上述 46 个字节减少到如下 31 个字节：

$$\square_2^{\square 0}[9]\parallel\Delta\square_2^{\square}[3]\parallel\square_3^{\square 0}[9,10]\parallel\square_4^{\square 0}[0,2,3,9,10,11]$$
$$\parallel\Delta\square_5^{\square}[0,1,2,3,5,7,8,9,10,12,13,14]$$
$$\parallel\square_7^{\square 0}[1,3,4,9,11,12]\parallel\square_8^{\square 0}[4,11]\parallel\Delta\square_8^{\square}[5]$$

事实上，由于差分存在关系$\Delta\square_2^{\square}[3]=\Delta\square_2^{\square}[12]=\Delta\square_2^{\square}[9]=\Delta\square_2^{\square}[10]$并且差分$\Delta\square_2^{\square}[8,11]=0$，故可计算$\Delta\square_3^{\square}[9,10]$的值。再根据$\square_3^{\square 0}[9,10]$的值，可计算出$\Delta\square_3^{\square}[9,10]$的值，进而可以推导出$\Delta\square_4^{\square}[0,2,3,9,10,11]$的值。又因为$\square_4^{\square 0}[0,2,3,9,10,11]$已知，所以$\Delta\square_4^{\square}[0,2,3,9,10,11]$的值可计算。类似地，可以计算$\Delta\square_5^{\square}[\square](\square\neq 6,14)$。根据$\square_3^{\square 0}[9,10]$和$\square_4^{\square 0}[0,2,3,9,10,11]$的值同时能够计算出$\square\square_3[1,8]$。

因为，$\Delta\square_5^{\square}[7]=\Delta\square_5^{\square}[4]\oplus\Delta\square_5^{\square}[5]\oplus\Delta\square_5^{\square}[6]=0$，$\Delta\square_5^{\square}[14]=\Delta\square_5^{\square}[12]\oplus\Delta\square_5^{\square}[13]\oplus\Delta\square_5^{\square}[15]=0$，并且$\Delta\square_5^{\square}[4]=0$，所以，得到$\Delta\square_5^{\square}[5]=\Delta\square_5^{\square}[6]$和$\Delta\square_5^{\square}[15]=\Delta\square_5^{\square}[12]\oplus\Delta\square_5^{\square}[13]$。由于$\Delta\square_5^{\square}[0\sim3,5,7\sim14]$已知，故可计算$\Delta\square_5^{\square}[0\sim5,7\sim13,15]$和$\Delta\square_6^{\square}[0\sim6,8\sim14,15]$的值。又因为$\Delta\square_5^{\square}[\square](\square\neq 6,14)$的值之前已经计算出，所以根据命题 1，可以计算出$\square_5^{\square}[0\sim5,7\sim13,15]$和$\square_5^{\square}[0\sim3,5\sim10,12\sim15]$。然后根据$\square_5^{\square 0}[\square](\square\neq 6,14)$和$\square_4^{\square 0}[0,2,3,9,10,11]$的值，计算$\square\square_4[0,1,6,8,9,14]$的值。

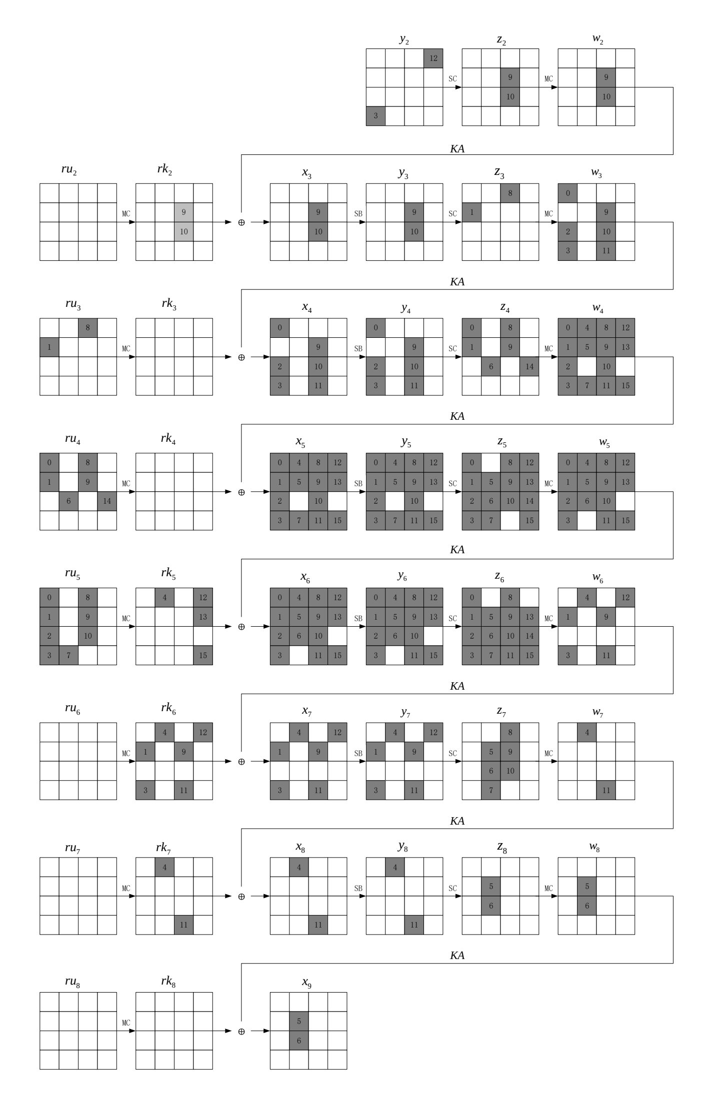


图 2　Midori128 算法 7 轮中间相遇区分器

因为$\Delta\square_8^{\square}[4,7]=\Delta\square_8^{\square}[4,7]=0$，所以$\Delta\square_8^{\square}[5]=\Delta\square_8^{\square}[6]$。因为$\Delta\square_8^{\square}[5]$和$\square_8^{\square 0}[4,11]$已知，故$\Delta\square_8^{\square}[4,11]$和$\Delta\square_7^{\square}[1,3,4,9,11,13]$可知。由于$\square_7^{\square 0}[1,3,4,9,11,12]$已知，故可计算出$\Delta\square_7^{\square}[1,3,4,9,11,13]$，并且根据输出差分的逆运算推导出$\Delta\square_6^{\square}[0\sim6,8\sim13,15]$。同样地，根据命题 1，由$\Delta\square_6^{\square}[0\sim6,8\sim14,15]$和$\Delta\square_6^{\square}[0\sim6,8\sim13,15]$可以得出$\square_6^{\square}[0\sim6,8\sim14,15]$。根据$\square_8^{\square 0}[4,11]$，$\square_7^{\square 0}[1,3,4,9,11,12]$值，$\square_6^{\square}[0\sim6,8\sim14,15]$可以计算出$\square\square_7[4,11]$，$\square\square_6[1,3,4,9,11,12]$。根据$\square_6^{\square}[0\sim6,8\sim14,15]$和$\square_5^{\square}[0\sim5,7\sim13,15]$可以计算出$\square\square_5[0,1,2,3,7,8,9,10,14]$和$\square\square_5[0,1,2,3,4,11,12,13,15]$。

因为，根据列混淆运算知道$\square\square_6[9]\oplus\square\square_6[10]=\square\square_6[9]\oplus\square\square_6[10]$，并且$\square\square_6[9,10]$可以由$\square\square_5[9,10]$推导出，所以可以计算出$\square\square_6[10]$。根据密钥编排规律，$\square\square_2[9,10]$可以从$\square\square_6[9,10]$推导出，并且已经知道$\square_3^{\square 0}[9,10]$，可以计算出$\square_2^{\square 0}[9,10]$。因为，

由列混淆运算知道$\square_2^{\square 0}[9]=\square_2^{\square 0}[8]\oplus\square_2^{\square 0}[10]\oplus\square_2^{\square 0}[11]$，$\square_2^{\square 0}[10]=\square_2^{\square 0}[8]\oplus\square_2^{\square 0}[9]\oplus\square_2^{\square 0}[11]$，所以$\square_2^{\square 0}[9]\oplus\square_2^{\square 0}[10]=\square_2^{\square 0}[9]\oplus\square_2^{\square 0}[10]$。知道$\square_2^{\square 0}[9]$，可以算出$\square_2^{\square 0}[10]$。根据密钥编排规律，$\square\square_3[1,8]$可以被$\square\square_5[1,8]$推导出，$\square\square_4[0,1,8,9,14]$可以被$\square\square_4[0,1,8,9,14]$推导出，$\square\square_7[4,11]$ 可以被$\square\square_5[4,11]$推导出，$\square\square_6[1,3,4,11,12]$可以被$\square\square_5[1,3,4,11,12]$和推导出。根据依赖密钥筛选技术，可以将 46 个字节降低到 31 个字节，共2^{136}种可能值。

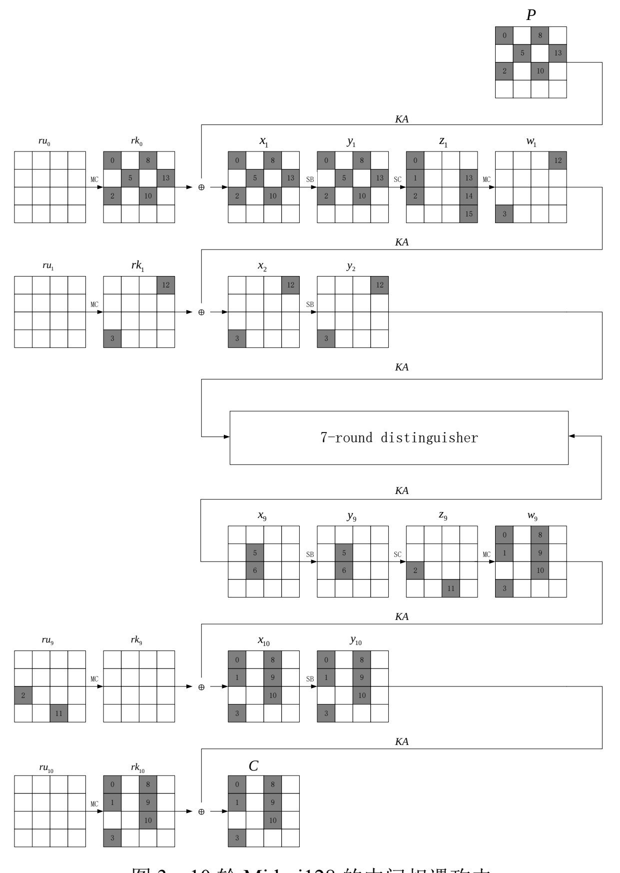


图 3　10 轮 Midori128 的中间相遇攻击

3.2　10 轮 Midori128 中间相遇攻击过程

基于 Midori128 算法的 7 轮中间相遇区分器，在区分器前端增加一轮，在区分器后端增加两轮来构造一个 10 轮的中间相遇攻击。如图 3 所示。攻击过程分为两个部分：预计算阶段和密钥筛选阶段。

1) 预计算阶段

在这一阶段，需要建立一个表$\square_0$来存储所有的有序序列$(\square_{\square\square\square}^{\square 1}\oplus\square_{\square\square\square}^{\square 0},\square_{\square\square\square}^{\square 2}\oplus\square_{\square\square\square}^{\square 0},\cdots,\square_{\square\square\square}^{\square 18}\oplus\square_{\square\square\square}^{\square 0})$，但在此之前，需要建立预计算表$\square_{\square}(1\le\square\le 6)$。

$\square_1$：猜测 72bit $\Delta\square_8[5]\parallel\square_8[4,11]\parallel\square_7[1,3,4,9,11,12]$所有可能值，则可计算出$\square_7[1,3,4,9,11,13],\square_7[4,11],\square_8[4,11]$，进而可得到$\square\square_7[4,11]=\square_7[4,11]\oplus\square_8[4,11]$。又因为$\Delta\square_8[5]=\Delta\square_8[6]$，故可计算出$\Delta\square_8[4,11]$和$\Delta\square_7[1,3,4,9,11,12]$，从而可计算出$\Delta\square_7[1,3,4,9,11,12]$和$\Delta\square_6[1,3,4,9,11,12]$。将$\Delta\square_6[1,3,4,9,11,12]\parallel\square_7[1,3,4,9,11,12]\parallel\square_8[4,11]$存储在表$\square_1$中，并通过$\square\square_7[4,11]$来检索，每一个检索值对应$2^{56}$个存储值。

$\square_2$：猜测 72bit $\Delta\square_2[3]\parallel\square_3[9,10]\parallel\square_4[0,2,3,9,10,11]$所有可能值，则可计算出$\square_3[1,8],\square_3[1,8]$，进而可以通过关系式得到$\square\square_3[1,8]=\square_3[1,8]\oplus\square_3[1,8]$。又因为$\Delta\square_2[3]=\Delta\square_2[12]$，故可计算出$\Delta\square_3[9,10]$和$\Delta\square_4[0,2,3,9,10,11]$，从而可计算出$\Delta\square_4[0,2,3,9,10,11]$和$\Delta\square_4[0,1,6,8,9,14]$。然后，将$\Delta\square_4[0,1,6,8,9,14]\parallel\square_4[0,2,3,9,10,11]\parallel\square_3[9,10]$存储在表$\square_2$中，并通过$\square\square_3[1,8]$来检索，每一个检索值对应$2^{56}$个存储值。

$\square_3$：猜测 112 bit $\Delta\square_4[1,6,8,9,14]\parallel\Delta\square_5[0,1,2,3]\parallel\Delta\square_6[1,3,4,11,12]$所有可能值，通过$\Delta\square_4[1,6,8,9,14]$可计算出$\Delta\square_5[0,5,10,15]$，通过$\Delta\square_5[0,1,2,3]$可计算出$\Delta\square_5[0,5,10,15]$和$\Delta\square_6[0,1,2,3]$，通过$\Delta\square_6[1,3,4,11,12]$可计算出$\Delta\square_6[0,1,2,3]$。根据命题 1，可以推导出$\square_5[0,5,10,15]$和$\square_6[0,1,2,3]$，进而计算出$\square_5[0,1,2,3]$和$\square_5[0,1,2,3]$并可以由关系式计算得到$\square\square_5[0,1,2,3]=\square_5[0,1,2,3]\oplus\square_5[0,1,2,3]$。将$\square_5[0,5,10,15]\parallel\square_6[0,1,2,3]$存储在表$\square_3$中，并通过$\square\square_5[0,1,2,3]\parallel\Delta\square_4[1,6,8,9,14]\parallel\Delta\square_6[1,3,4,11,12]$来检索，每一个检索值有 1 个存储值。

$\square_4$：猜测 80bit$\Delta\square_4[0,6,8,9]\parallel\Delta\square_5[5,7]\parallel\Delta\square_6[1,3,4,9]$所有可能值。因为$\Delta\square_5[7]=0$，所以$\Delta\square_5[5]=\Delta\square_5[6]$。通过$\Delta\square_4[0,6,8,9]$可计算出$\Delta\square_5[1,4,11]$，通过$\Delta\square_5[5,6,7]$可计算出$\Delta\square_5[1,4,11]$和$\Delta\square_6[4,5,6]$，通过$\Delta\square_6[1,3,4,9]$可计算出$\Delta\square_6[4,5,6]$。根据命题 1，可以推导出$\square_5[1,4,11]$和$\square_6[4,5,6]$，进而计算出$\square_5[5,6,7],\square_5[4]$和$\square_5[7]$并能得到$\square\square_5[7]=\square_5[7]\oplus\square_5[7]$，$\square\square_5[4]=\square_5[4]\oplus\square_6[4]$。将$\square_5[1,4,11]\parallel\square_6[4,5,6]$存储在表$\square_4$中，并通过$\square\square_5[7]\parallel\square\square_5[4]\parallel\Delta\square_4[0,6,8,9]\parallel\Delta\square_6[1,3,4,9]$来检索，每一个检索值有 1 个存储值。

$\square_5$：猜测 96 bit$\Delta\square_4[0,1,8,14]\parallel\Delta\square_5[8,9,10]\parallel\Delta\square_6[3,4,9,11,12]$所有可能值，通过$\Delta\square_4[0,1,8,14]$可计算出$\Delta\square_5[3,9,12]$，通过$\Delta\square_5[8,9,10]$可计算出$\Delta\square_5[3,9,12]$和$\Delta\square_6[8,9,10,11]$，通过$\Delta\square_6[3,4,9,11,12]$可计算出$\Delta\square_6[8,9,10,11]$。根据命题 1，可以推导出$\square_5[3,9,12]$和$\square_6[8,9,10,11]$，进而计算出$\square_5[8,9,10]$和$\square_5[8,9,10,11]$。然后，可以计算出$\square\square_5[8,9,10]=\square_5[8,9,10]\oplus\square_5[8,9,10]$。将$\square_5[3,9,12]\parallel\square_6[8,9,10,11]$存储在表$\square_5$中，并通过$\square\square_5[8,9,10]\parallel\Delta\square_4[0,1,8,14]\parallel\Delta\square_6[3,4,9,11,12]$来检索，每一个检索值有 1 个存储值。

$\square_6$：猜测 80 bit $\Delta\square_4[0,1,6,9,14]\parallel\Delta\square_5[12,13,14]\parallel\Delta\square_6[1,9,11,12]$所有可能值。因为$\Delta\square_5[14]=0$，所以$\Delta\square_5[15]=\Delta\square_5[12]\oplus\Delta\square_5[13]$。通过$\Delta\square_4[0,1,6,9,14]$可计算出$\Delta\square_5[2,7,8,13]$，通过$\Delta\square_5[12,13,14]$可计算出$\Delta\square_5[2,7,8,13]$和$\Delta\square_6[12,13,15]$，通过$\Delta\square_6[1,9,11,12]$可计算出$\Delta\square_6[12,13,15]$。根据命题 1，可以推导出$\square_5[2,7,8,13]$和$\square_6[12,13,15]$，进而计算出$\square_5[12,13,15]$，并且能够得到$\square\square_5[12,13,15]=\square_5[12,13,15]\oplus\square_6[12,13,15]$。然后，将$\square_5[2,7,8,13]\parallel\square_6[12,13,15]$存储在表$\square_6$中，并通过$\square\square_5[12,13,15]\parallel\Delta\square_4[0,1,6,9,14]\parallel\Delta\square_6[1,9,11,12]$来检索，每一个检索值有 1 个存储值。

$\square_0$：猜测 96bit$\square\square_5[0,1,2,3,7,8,9,10]\parallel\square\square_5[4,12,13,15]$所有可能值，做如下运算。

根据密钥编排计划，通过$\square\square_4[4]$和$\square\square_5[8,9,10]$可以得到$\square\square_7[4,11]$。然后通过$\square\square_7[4,11]$搜索表$\square_1$，得到值

$\Delta\square_6[1,3,4,9,11,12] \| \square_7[1,3,4,9,11,12] \| \square_8[4,11]$。同样，通过$\square\square_5[1,8]$，可以得到值$\square\square_3[1,8]$，搜索表$\square_2$，得到$\Delta\square_4[0,1,6,8,9,14] \| \square_4[0,2,3,9,10,11] \| \square_3[9,10]$。

对于所有在以上过程中已经搜索得到的2^{112}个值$\Delta\square_6[1,3,4,9,11,12] \| \square_7[1,3,4,9,11,12] \| \square_8[4,11] \|$ $\Delta\square_4[0,1,6,8,9,14] \| \square_3[9,10] \| \square_4[0,2,3,9,10,11]$，检测表$\square_3$可以得到值$\square_5[0,5,10,15] \| \square_6[0,1,2,3]$。因为$\square\square_5[7]$和$\square\square_5[4]$已知，通过$\square\square_5[7] \| \square\square_5[4] \| \Delta\square_4[0,6,8,9] \| \Delta\square_6[1,3,4,9]$检测表$\square_4$得到值$\square_5[4,11,1] \| \square_6[4,5,6]$。同样地，通过$\square\square_5[8,9,10] \| \Delta\square_4[0,1,8,14] \| \Delta\square_6[3,4,9,11,12]$检测表$\square_5$得到值$\square_5[3,9,12] \| \square_6[8,9,10,11]$。通过$\square\square_5[12,13,15] \| \Delta\square_4[0,1,6,9,14] \| \Delta\square_6[1,9,11,12]$检测表$\square_6$得到值$\square_5[7,13,2,8] \| \square_6[12,13,15]$。

对于从表$\square_3$，$\square_4$，$\square_5$，$\square_6$检测到的值$\square_5[0\sim5,7\sim13,15] \| \square_4[0,2,3,9,10,11]$，可以计算出$\square_4[0,1,6,8,9,14]$和$\square_4[0,1,6,8,9,14]$，从而可以得到$\square\square_4[0,1,6,8,9,14] = \square_4[0,1,6,8,9,14] \oplus \square_4[0,1,6,8,9,14]$。同样，通过$\square_6[0\sim6,8\sim13,15] \| \square_7[1,3,4,9,11,12]$可以计算出$\square\square_6[1,3,4,9,11,12]$。根据密钥编排计划，通过$\square\square_4[0,1,8,9,14]$和$\square\square_6[1,3,4,11,12]$可以计算出密钥$\square\square_5[0,1,8,9,14]$和$\square\square_5[1,3,4,11,12]$。将该计算出的值与原来的猜测密钥$\square\square_5[0,1,8,9,14] \| \square\square_5[1,3,4,11,12]$进行对比。如果两个值相等，则保留该一系列的值$\square_3[9,10] \| \square_4[0,2,3,9,10,11] \| \square_5[\square](\square \neq 6,14) \| \square_6[\square](\square \neq 7,14) \| \square\square_6[1,3,4,9,11,13] \| \square\square_7[4,11]$。一共有$2^{112}$个检索值，但是每个值被保留的概率为$2^{-80}$，所以最终有$2^{32}$个值保留下来。

对于所有2^{32}个值，根据密钥编排规律和已经计算出的密钥值可以推导出$\square\square_2[9,10]$并且计算$\square_2[9,10]$。根据列混淆运算，可以知道$\square_2[9] \oplus \square_2[10] = \square_2[9] \oplus \square_2[10]$，所以对于$2^8$个值$\square_2[9]$，可以计算出$\square_2[10]$并此推导出$\square_2[3,12]$。由此，得到所有 46 字节的参数。通过这些所计算的参数计算有序序列$(\square^{\square_1}_{\square\square\square} \oplus \square^{\square_0}_{\square\square\square}, \square^{\square_2}_{\square\square\square} \oplus \square^{\square_0}_{\square\square\square}, \cdots, \square^{\square_{18}}_{\square\square\square} \oplus \square^{\square_0}_{\square\square\square})$，并且将 56bit$\square\square_5[0,1,2,3,12,13] \| \square\square_6[9]$的值一并保存。

2) 密钥筛选阶段

在对密钥进行筛选之前，需要预计算 4 个预计算表$T_i(7 \leq i \leq 10)$，具体如下：

$\square_7$：猜测 72bit$\Delta\square_9[5] \| \Delta\square_9[2,11] \| \square_{10}[0,1,3,8,9,10]$所有可能值，可以计算出$\Delta\square_9[5,6]$，$\square_9[2,11]$。因为$\Delta\square_9[5] = \Delta\square_9[6]$，根据命题 1，可得到$\square_9[5,6]$，$\square_9[2,11]$。然后可以计算出$\square\square_9[2,11] = \square_9[2,11] \oplus \square_9[2,11]$。将$\square_9[5,6] \| \Delta\square_9[5]$存储在表$\square_7$中，并通过$\square\square_9[2,11] \| \square_{10}[0,1,3,8,9,10] \| \Delta\square_9[2,11]$来检索，每一个检索值对应$2^{-8}$个存储值。

$\square_8$：猜测 56bit$\Delta P[0,5,10] \| P[0,5,10] \| \Delta\square_1[3]$所有可能值，可以计算出$\Delta\square_1[0,5,10]$，$\Delta\square_1[0,5,10]$。根据命题 1，可得到$\square_1[0,5,10]$和$\square_1[0,5,10]$，然后计算出$\square\square_0[0,5,10](= P[0,5,10] \oplus \square_1[0,5,10])$和$\square_1[3]$。将$\square_1[3] \| \Delta\square_1[3] \| \square\square_0[5]$存储在表$\square_8$中，并通过$\square\square_0[0,10] \| \Delta P[0,5,10] \| P[0,5,10]$来检索，每一个检索值对应$2^{-8}$个存储值。

$\square_9$：猜测 56bit$\Delta P[2,8,13] \| P[2,8,13] \| \Delta\square_1[12]$所有可能值，计算出$\Delta\square_1[2,8,13]$，$\Delta\square_1[2,8,13]$。根据命题 1，可得到$\square_1[2,8,13]$和$\square_1[2,8,13]$，然后计算出$\square\square_0[2,8,13] = P[2,8,13] \oplus \square_1[2,8,13]$和$\square_1[12]$。将$\square_1[12] \| \Delta\square_1[12] \| \square\square_0[2,13]$存储在表$\square_9$中，并通过$\square\square_0[8] \| \Delta P[2,8,13] \| P[2,8,13]$来检索，每一个检索值对应 1 个存储值。

$\square_{10}$：猜测 40 bit $\Delta\square_1[3,12] \| \square_1[3,12] \| \Delta\square_2[3]$所有可能值，计算$\Delta\square_2[3,12]$。因为$\Delta\square_2[3] = \Delta\square_{12}[12]$，根据命题 1，可以得到$\square_2[3,12]$和$\square_2[3,12]$，再计算$\square\square_1[3,12] = \square_1[3,12] \oplus \square_2[3,12]$。然后，将$\square_2[3,12] \| \square\square_1[12]$存储在表$\square_{10}$中，通过$\square\square_1[3] \| \Delta\square_1[3,12] \| \square_1[3,12]$来检索，每一个检索值对应 1 个存储值。

接下来，在密钥筛选阶段，首先要找到一对正确对满足截断差分链；再通过这一个正确对计算有序序列$(\square^{\square_1}_{\square\square\square} \oplus \square^{\square_0}_{\square\square\square}, \square^{\square_2}_{\square\square\square} \oplus \square^{\square_0}_{\square\square\square}, \cdots, \square^{\square_{18}}_{\square\square\square} \oplus \square^{\square_0}_{\square\square\square})$，检测其是否满足表$\square_0$，最后利用表$\square_0$中存储的密钥再次判断计算出的密钥$\square\square_5[0,1,2,3,12,13] \| \square\square_6[9]$是否满足表$\square_0$，并筛选出正确的密钥。

假设明文$P[0,2,5,8,10,13]$取任意值，其余 10 个字节取固定值，这样形成了一个结构，共有2^{48}个明文值，可以形成$2^{48+47} = 2^{95}$个明密文对。选取$P[1,3,4,6,7,9,11,12,14,15]$的$2^{65}$个不同的值，也即选取$2^{113}$个明文，形成$2^{160}$个明密文对。因为一个明文对能够满足 7 轮截断差分链的概率为$2^{(1-6+1-16)\times8} = 2^{-160}$，所以在每一个猜测密钥下，平均有一对能够满足截断差分链。加密这2^{113}个明文得到密文，筛选出满足密文差分为$\Delta C[2,4,5,6,7,11,12,13,14,15] = 0$的明密文，共$2^{160-10\times8} = 2^{80}$明密文对。

对于这2^{80}明密文对，做如下操作：

a) 猜测$\Delta\square_9[2,11]$的所有可能值，可以计算出$\Delta\square_{10}[0,1,3,8,9,10]$，根据命题 1，通过$\Delta\square_{10}[0,1,3,8,9,10]$和$\Delta C[0,1,3,8,9,10]$可以得到$\square_{10}[0,1,3,8,9,10]$，计算出$\square\square_{10}[0,1,3,8,9,10]$，从而可以得到$\square\square_{10}[0,1,3,8,9,10]$的值，即$\square\square_9[2,11]$的值。一共有$2^{16}$个值$\Delta\square_9[2,11]$，所以推导出$2^{16}$个$\Delta\square_9[2,11] \| \square\square_{10}[0,1,3,8,9,10] \| \square\square_9[2,11] \| \square_{10}[0,1,3,8,9,10]$，对于这些值搜索表$\square_7$，得到$2^{-8}$个检索值$\square_9[5,6] \| \Delta\square_9[5]$，则有$2^8$个值满足截断差分链后端的差分。

b) 对于2^8个值$\square\square_{10}[0,1,3,8,9,10] \| \square\square_9[2,11]$，根据密钥编排计划，可以计算出$\square\square_0[0,10]$。利用$\square\square_0[0,10] \| \Delta P[0,5,10] \| P[0,5,10]$搜索表$\square_8$，平均来讲，对于每一个检索得到$2^{-8}$个值$\square_1[3] \| \Delta\square_1[3] \| \square\square_0[5]$。所以，平均来说一共有 1 个值$\square\square_{10}[0,1,3,8,9,10] \| \square\square_9[2,11] \| \square_1[3] \| \Delta\square_1[3] \| \square\square_0[5]$被保留。

c) 对于这一个值$\square\square_{10}[0,1,3,8,9,10] \| \square\square_9[2,11] \| \square_1[3] \| \Delta\square_1[3] \| \square\square_0[5]$，根据密钥编排计划，可以计算出$\square\square_0[8]$，搜索表$\square_9$，平均来，可以得到 1 个检索值$\square_1[12] \| \Delta\square_1[12] \| \square\square_0[2,13]$。

d) 对于以上步骤中已得到的值$\square\square_{10}[0,1,3,8,9,10] \| \square\square_9[2,11] \| \square_1[3] \| \Delta\square_1[3] \| \square\square_0[2,5,13] \| \square_1[12] \| \Delta\square_1[12]$，可以通过$\square\square_{10}[3]$计算$\square\square_1[3]$，并且搜索表$\square_{10}$，得到 1 个检索值$\square_2[3,12] \| \square\square_1[12]$。最终对于每一个明密文对来说，平均只有 1 个可能值$\square\square_{10}[0,1,3,8,9,10] \| \square\square_9[2,11] \| \square\square_0[0,2,5,8,10,13] \|$

$\square\square_1[3,12] \parallel \square_2[3,12]$。

e） 根据有序序列的差分值，改变$\square_2[3,12]$的值得到一个序列为$\{\square_2^{\square_0},\square_2^{\square_1},\cdots,\square_2^{\square_{18}}\}$。根据检测到的密钥将这一序列$\{\square_2^{\square_0},\square_2^{\square_1},\cdots,\square_2^{\square_{18}}\}$推导出明文值$\{P^{\square_0},P^{\square_1},\cdots,P^{\square_{18}}\}$，并且计算出相应密文$\{C^{\square_0},\square^{\square_1},\cdots,C^{\square_{18}}\}$。同样，用所求的密钥解密得到$(\square_{\square\square\square}^{\square_1}\oplus\square_{\square\square\square}^{\square_0},\square_{\square\square\square}^{\square_2}\oplus\square_{\square\square\square}^{\square_0},\cdots,\square_{\square\square\square}^{\square_{18}}\oplus\square_{\square\square\square}^{\square_0})$。若所计算的序列值不在表$\square_0$中，则排除相应密钥。若序列值存在表$\square_0$中，则再次判断密钥值$\square\square_0[2,13] \parallel \square\square_1[3,12] \parallel \square\square_{10}[0,1,3,9]$是否能够推导出表$\square_0$中的$\square\square_5[0,1,2,3,12,13] \parallel \square\square_6[9]$。若能，则此密钥被保留，保留下来的概率为$2^{(17-18)\times 8-56} = 2^{-64}$。

将剩下2^{16}个密钥和 6 个字节的未猜测密钥进行穷举搜索。

3.3 复杂度分析

攻击的复杂度分析分为预计算阶段和密钥筛选阶段两部分。在预计算阶段，构造T_0需要对 19 个信息值进行2^{136}次 7 轮的部分的加密，所以时间复杂度为$2^{136}\times 19\times 44/160=2^{138}$次 10 轮 Midori128 加密，存储复杂度为$2^{136}\times 19\times 8\times 2^{-7}=2^{136}$个 128-bit 块，且它们决定了预计算阶段的复杂度。在筛选密钥阶段，首先需要加密2^{113}个选择明文来得到相应的密文，并筛选出符合密文差分的明密文对。所以时间复杂度为2^{113}次 10 轮 Midori128 加密，数据复杂度2^{113}个明文。然后对剩余的2^{80}个明密文对以及相应的 19 个信息值部分加密来判断有序序列是否属于表T_0中。时间复杂度为$2^{80}\times 19\times 16/160=2^{81}$次 10 轮 Midori128 加密。因此密钥筛选阶段的数据复杂度、时间复杂度和存储复杂度分别为2^{113}个选择明文，2^{81}次 10 轮 Midori128 加密和2^{82}128bit 块。综上，整个攻击的数据复杂度、时间复杂度和存储复杂度分别为2^{113}个选择明文、2^{138}次 10 轮 Midori128 加密和2^{136}个 128-bit 块。

此外，为了优化算法的复杂度，可以对攻击过程进行数据、时间和存储复杂度折中，令折中的系数$\alpha=2^{12}$，所以预计算阶段的时间复杂度为2^{126}次 10 轮 Midori128 加密，存储复杂度为2^{124}128-bit 块。而折中后密钥筛选阶段的数据复杂度为2^{125}个选择明文，时间复杂度为$2^{125}+2^{93}\approx 2^{125}$次 10 轮 Midori128 加密，存储复杂度为2^{94}128-bit 块。因此，整个攻击时间复杂度为$2^{126}+2^{125}\approx 2^{126.5}$次 10 轮 Midori128 加密，数据复杂度为2^{125}个明文，存储复杂度为$2^{124}+2^{94}\approx 2^{124}$128-bit 块。

更进一步，由于预计算阶段和密钥筛选阶段分别需要猜测的密钥包含$ru_5[0,1,2,3]$和$rk_{10}[0,1,3] /\!/ rk_0[2]$，可以根据密钥计划，利用它们的关系做弱密钥攻击，即对应于每个$ru_5[0,1,2,3]$的固定值，建立T_0的子表T_0^*，因此存储复杂度可以降低2^{-32}倍，故建立T_0^*的存储复杂度在预计算阶段不占主要的，而建立预计算表T_3所需的存储复杂度将占主导地位，因此整个攻击的存储复杂度可降为$2^{112-7}=2^{105}$128-bit。

分组长度为 128bit 的 Midori128 算法结构与 AES 类似，使用了 16 个 8bit 的 S 盒和分支数为 4 的类 MDS 矩阵，但 Midori128 用按字节置换操作替换 AES 的行移位操作，使得算法的扩散性和混淆性更好，直接影响到中间相遇攻击区分器构造，提高了 Midori128 算法抵抗中间相遇攻击的能力。分析结果表明，在结合差分枚举和依赖密钥筛选的技巧下，10 轮的 Midori128 算法不能够很好的抵抗中间相遇攻击，而更高轮数的 Midori128 算法目前可以抵抗中间相遇攻击。

4 结束语

本文主要研究了 Midori128 算法对抗中间相遇攻击的安全性分析。首先是利用利用差分枚举技巧和依赖密钥筛选技巧来构造一个 7 轮中间相遇区分器，并且在这个区分器前端增加一轮，后端增加两轮，实现了对 Midori128 算法 10 轮抵抗中间相遇攻击。最后运用时空复杂度折中和弱密钥分别降低了攻击的时间复杂度和存储复杂度。因此，整个攻击的数据复杂度2^{125}选择明文，时间复杂度为$2^{126.5}$10 轮 Midori128 加密，存储复杂度2^{105}128-bit 块。这是第一个对 Midori-128 抵抗中间相遇的分析。

参考文献：

[1] Banik S, Bogdanov A, Isobe T, et al. Midori: a block cipher for low energy [C]// Advances in Cryptology-ASIACRYPT 2015. Berlin: Springer, 2015: 411-436.

[2] Dong Xiaoyang, Shen Yanzhao. Cryptanalysis of reduced-round Midori64 block cipher [EB/OL]. http://eprint. iacr. org/2016/676. pdf.

[3] Chen Zhan, Bi Wenquan, Wang Xiaoyun. Impossible differential cryptanalysis of midori64 [EB/OL]. http://eprint. iacr. org/2016/535. pdf.

[4] Takahashi Y. Higher-order differential attack on the round-reduced variants of the block cipher Midori64, IEICE Tech. Rep. [R]. 2016: 159-164.

[5] Lin Li, Wu Wenling. Meet-in-the-middle attacks on reduced-round Midori-64 [EB/OL]. http://eprint. iacr. org/2015/1165. pdf.

[6] Guo Jian, Jean J, Nikoli'c I, et al. Invariant subspace attack against full midori64 [EB/OL]. http://eprint. iacr. org/2015/1189. pdf.

[7] Chen Zhan, Chen Huaifeng, Wang Xiaoyun. Cryptanalysis of Midori128 using impossible differential techniques [C]// Proc of Information Security Practice and Experience. Berlin: Springer, 2016: 1-12.

[8] Bi Wenquan, Li Zheng, Dong Xiaoyang. Impossible differential attack on Midori128 using rebound-like technique [EB/OL]. http://eprint. iacr. org/2017/286. pdf.

[9] Tolba M, Abdelkhalek A, Youssef A M. Truncated and multiple differential cryptanalysis of reduced round Midori128 [C]// Proc of Information Security. Berlin: Springer, 2016: 3-17.

[10] Gérault D, Lafourcade P. Related-key cryptanalysis of midori [C]// Proc of Cryptology-INDOCRYPT 2016. Berlin: Springer, 2016: 287-304.

[11] Li Rongjia, Jin Chenhui. Meet-in-the-middle attacks on 10-round AES-256 [J]. Designs, Codes and Cryptography, 2016, 80 (3): 459-471.

[12] Diffie W, Hellman M E. Special feature exhaustive cryptanalysis of the NBS data encryption standard [J]. Computer, 1977, 10 (6): 74-84.

[13] Li Leibo, Jia K, Wang Xiaoyun, et al. Improved meet-in-the-middle attacks on AES-192 and PEINCE [EB/OL]. http://eprint. iacr. org/2013/573. pdf.

[14] Daemen J, Rijmen V. The design of Rijndael: AES-the advanced encryption standard [M]. Berlin: Springer, 2002.

[15] Gilbert H, Minier M. A collision attack on 7 rounds of Rijndael [C]// Proc of the 3rd AES Candidate Conference. Berlin: Springer, 2000: 230-241.

[16] Demirci H, Selçuk A A. A meet-in-the-middle attack on 8-round AES [C]// Proc of Fast Software Encryption. Berlin: Springer, 2008: 116–126.

[17] Dunkelman O, Keller N, Shamir A. Improved single-key attacks on 8-round AES-192 and AES-256 [C]// Proc of Advances in Cryptology-ASIACRYPT 2010. Berlin: Springer, 2010: 158-176.

[18] Li Leibo, Jia K, Wang Xiaoyun. Improved single-key attacks on 9-round AES-192//256 [C]// Pro of Fast Software Encryption. Berlin: Springer, 2015: 127-146.

[19] Guo Jian, Peyprin T, Poschmann A, et al. The LED block cipher [C]// Proc of Cryptographic Hardware and Embedded systems. Berlin: Springer, 2011: 326-341.

[20] Li Leibo, Jia K, Wang Xiaoyun, et al. Meet-in-the-middle technique for truncated differential and its applications to CLEFIA and Camellia [C]// Pro of Fast Software Encryption. Berlin: Springer, 2015: 48-70.

[21] 汪艳凤，吴文玲. 分组密码 TWINE 的中间相遇攻击 [J]. 软件学报, 2015, 26 (10): 2684-2695.

[22] 郑雅菲，吴文玲. LBlock 算法的改进中间相遇攻击 [J]. 计算机学报, 2017, 40 (5): 1080-1091.